Towards Enabling RAN as a Service - The Extensible Virtualisation Layer

Joao F. Santos*, Maicon Kist*†, Jonathan van de Belt*, Juergen Rochol†, Luiz A. DaSilva*
*CONNECT - Trinity College Dublin, Ireland
{facocalj, kistm, vandebej, dasilval}@tcd.ie
†Federal University of Rio Grande do Sul, Brazil
{mkist, juergen}@inf.ufrgs.br

***Abstract*—Network slicing is one of the key enabling techniques for 5G, allowing Mobile Network Operators (MNOs) to support services with diverging requirements on top of their infrastructure. The MNOs should be able to offer network slices as a service and provide customisable and independent virtual networks to verticals. The slicing of an end-to-end (E2E) mobile network is divided into Core Network (CN) slicing, and Radio Access Network (RAN) slicing. In this paper, we assess the requirements for using radio hypervisors to enable RAN as a Service (RANaaS). We evaluate the current state-of-the-art on radio virtualisation with respect to these requirements and identify the missing features. Then, we present the eXtensible Virtualisation Layer (XVL), a software layer that provides the missing functionality for enabling RANaaS and can be added on top of existing radio hypervisors. We outline XVL's architecture and design choices, as well as evaluate its performance in terms of the delay to provision virtual radios, the delay introduced to forward IQ samples, and the computational overhead. Our results show that XVL enables leveraging existing radio hypervisors to support RANaaS.**

***Keywords*—Network Slicing, Radio Access Network, Virtualisation, RAN as a Service, Radio Hypervisor**

I. Introduction

In contrast to previous generations of mobile networks, 5G is being envisioned from the very beginning to support a variety of different services, e.g., Enhanced Vehicular-to-Everything (V2X), massive Internet of Things (IoT), and industrial automation [1]. In order to cope with such diverse network requirements, 3GPP introduced the concept of network slicing for 5G, which proposes the partitioning of the physical network infrastructure of a Mobile Network Operator (MNO) into independent logical networks, known as Network Slices (NSs) [2]. Each NS operates as a separate end-to-end (E2E) network, which can be individually tailored and configured for different purposes [3]. Network slicing creates new business models for the mobile networks market by enabling new ways for an MNO to monetise its infrastructure [4]. In this way, MNOs can offer NS as a Service (NSaaS) to provide independent and isolated virtual networks to verticals, running on top of a shared physical network infrastructure [2].

The 3GPP defines E2E network slicing as the combination of Core Network (CN) slicing and Radio Access Network (RAN) slicing [5]. Both network segments can be independently orchestrated, sliced, and combined for creating E2E NSs [6]. As a consequence, the NSaaS paradigm is the combination of CN as a Service (CNaaS) [7] and RAN as a Service (RANaaS) [8]. This separation grants MNOs the flexibility to offer: CN slices, where each vertical can define its own authentication schemes, mobile and session management, whilst sharing a common RAN among all clients; RAN slices, where each vertical can specify the sub-carrier spacing, cyclic prefix length, and handover threshold, whilst sharing a common CN among all clients; or full E2E NSs, where verticals can define both their own RAN and CN [4].

The slicing process is performed by an entity known as a hypervisor, which is tailored for virtualising a particular type of resource, e.g., network hypervisors use network resources for creating virtual networks, and radio hypervisors use radio resources for creating virtual radios. Currently, the virtualisation of the CN is a mature area, with network hypervisors, e.g., FlowVisor [9] and OpenVirteX [10], being adopted in production networks. On the other hand, RAN virtualisation is a new research topic. There are examples of open-source radio hypervisors available in the literature, which can create virtual radios to realise a RAN slice, using either low-level resources, e.g., time and frequency; or high-level resources, e.g., Physical Resource Blocks (PRBs) and frames [11]. Albeit paving the way to allow RAN slicing, these prototypes only focus on enabling multiple radio slices to coexist on top of a single physical hardware, without considering the additional requirements for RANaaS. For instance, the lack of a radio broker for verticals to communicate for querying the available radio resources, requesting RAN slices, and assessing the performance of their RAN deployment [12], prevents the realisation of RANaaS.

In this paper, we address the functionality that is necessary for using radio hypervisors as enablers for RANaaS. We have developed the eXtensible Virtualisation Layer (XVL), a software layer that implements such functionality and that can be added on top of existing radio hypervisors. XVL allows the negotiation of the available resources with verticals, the creation of virtual radios on demand, and it uses cross-platform communication libraries, making it easy to integrate with different programming languages and Software-defined Radio (SDR) platforms. We have integrated XVL with a radio hypervisor, validated its ability to serve as an enabler for RANaaS on top of current radio hypervisors, and evaluated XVL's performance in different scenarios.

II. Radio Virtualisation for RAN as a Service

One of the initial works on radio virtualisation, Virtual Radio [12] argued for how radio virtualisation could mitigate

the slow development cycle in the wireless network industry, advocating for the deployment of software-based base stations with a tailored amount of radio resources and protocol stack. Ultimately, the work of [12] proposes the use of radio virtualisation for enabling the deployment of virtual wireless networks on demand, years before the term RANaaS was coined [8]. The majority of the following works on radio virtualisation focused on creating radio hypervisors to enable radio slicing, and ensuring isolation at a radio resource level. The next natural step is to move from performing radio slicing with radio hypervisors, to leveraging the capabilities of current radio hypervisors for enabling the RANaaS paradigm. Inspired by the initial concepts in [12], and further works on virtualisation of wireless networks [11], [13]–[15], we devised a list of required features for a radio hypervisor to be able to support RANaaS.

A. Requirements for Supporting RANaaS

The providers of RANaaS, e.g., network operators or infrastructure providers, can have different radio resources, over different geographical locations, and may employ different types of pricing per band or duration, based on auctions or any other business models [16]. Hence, the customers of RANaaS, e.g., Mobile Virtual Network Operators (MVNOs) or verticals, must be able to discover the available radio resources and negotiate their use [12]. The interaction between radio hypervisors and customers requires an interface for receiving remote queries and requests. Such an interface must have a well-defined syntax for the communication, and a clear abstraction for describing the available radio resources, location and cost. Upon successful reservation of radio resources, the radio hypervisor must be able to instantiate virtual radios on demand, as customers may start and halt the operation of RAN slices at will.

A radio hypervisor should be platform agnostic and support the creation of virtual radios with any Radio Access Technologies (RATs). However, different RATs not only possess different definitions of resources but also use the medium in various manners [14]. For example, LTE and WiMax allocate PRBs to users and transmit continuously (downlink), whereas WiFi and Bluetooth allocate entire frames to users and change transmission interval according to the traffic. Therefore, it is not possible to create a generalised resource grid for all possible RATs and treat radio virtualisation as a scheduling problem [12]. Instead, it should be treated as a multiplexing problem, where: (*i*) the virtual radios use low-level radio resources, bellow the PRB/frame level, e.g., centre frequency, bandwidth, timeslot; (*ii*) each RAN slice has complete control over their resources; (*iii*) the radio hypervisor multiplexes the virtual radios without compromising the operation of their RATs [11] [17]. In addition, virtual radios must be isolated both regarding their radio resources to prevent interference between virtual radios; and their computational resources, e.g., different instances or processes, to ensure independence between virtual radios.

The use of a radio hypervisor for deploying RAN slices introduces a new layer of complexity and another point of failure. The radio resources previously dedicated to realising a single RAT now constitute a pool of resources shared amongst a number of virtual radios, which in turn, are used to provide the RANaaS. Hence, customers should be able to assess the performance of their virtual radios for ensuring the proper operation of their RAN deployment, and monitoring the established Service Level Agreement (SLA) [15]. For instance, customers should be able to query Key Performance Indicators (KPIs) of their virtual radios, e.g., buffer sizes, processing delay, and SINR degradation. Ideally, the prototype implementation of the radio hypervisors should be made available to the customers, for transparency and security [13], and to the research community, allowing further research and development.

Based on the aforementioned requirements, we summarise the necessary features for a radio hypervisor to support RANaaS:

- Resource Negotiation: to possess an interface for querying and requesting the available resources, describing the radio resources, e.g., in terms of centre -frequency, bandwidth, and the virtual radios, e.g., owner, transmitter and/or receiver capabilities.
- Dynamic Allocation: to allow the creation and destruction of virtual radios on demand, without interrupting the operation of the radio hypervisor or other virtual radios.
- Technology Agnostic: to employ a multiplexing technique that supports different RATs, and does not distort their waveforms, nor limit their medium access techniques.
- Radio Resource Isolation: to allocate non-overlapping radio resources to different virtual radios, and prevent interference between virtual radios, e.g., through filtering, guard bands, or guard times.
- Virtual Radio Independence: to ensure that virtual radio instances cannot affect the operation and performance of each other, even in case of a malfunctioning or misbehaving virtual radio.
- Service Monitoring: to collect and provide information about the performance of individual virtual radios, e.g., in terms of buffer sizes, and the introduced delay and SINR degradation.
- Open-Source Prototype: to grant access to source code of their implementation to customers and academia.

B. State of the Art on Radio Virtualisation

We have evaluated some notable examples of radio hypervisors in the literature with respect to the requirements described in Section II-A, and Table I summarises the results of our analysis. The majority of research efforts on RAN slicing and radio virtualisation either focus only on particular technologies, or do not make available their implementation [14].

One of the few works that are platform agnostic and provides an actual open-source implementation is HyDRA [17], a radio hypervisor for SDRs developed on top of GNU Radio [22]. HyDRA virtualises the Radio Frequency (RF) front-end of a single SDR to create virtual RF front-ends.

Radio Hypervisor	Resource Negotiation	Dynamic Allocation	Technology Agnostic	Radio Resource Isolation	Virtual Radio Independence	Service Monitoring	Open-Source Prototype
NVS [18]	-	-	-	✓	-	-	-
Virtual WiFi [19]	-	✓	-	✓	-	-	-
Orion [20]	✓	✓	-	✓	✓	-	-
SVL [21]	-	-	✓	✓	-	-	-
HyDRA [17]	-	-	✓	✓	-	-	✓
XVL [this work]	✓	✓	✓	✓	✓	✓	✓

TABLE I: Qualitative evaluation of the features necessary for radio hypervisors to support RANaaS.

Similar to the work of [21], HyDRA uses an FFT-based technique to slice the bandwidth of a radio's RF front-end into virtual RF front-ends. Each virtual radio can realise any RAT using its own virtual RF front-end. However, if any virtual radio delays transmitting/receiving or crashes, it will halt the operation of the radio hypervisor and all other virtual radios. Moreover, the number of virtual radios and their resource allocation, i.e., the amount of spectrum for each virtual RF front-end, must be set up before the operation of the radio hypervisor, as it is not possible to (de)allocate a virtual radio without interrupting the radio hypervisor's operation. Nonetheless, HyDRA excels at radio virtualisation for allowing multiple heterogeneous virtual radios to share the same underlying physical radio hardware.

III. Extensible Virtualisation Layer

In order to tackle the gaps and missing features detailed in Section II, we have developed XVL, a cross-platform software layer that sits on top of existing radio hypervisors. XVL works as a wrapper, leveraging the underlying radio hypervisor's capabilities and providing it with the missing functionality for supporting RANaaS. This modular design allows the radio hypervisor to focus on the radio virtualisation, offloading to XVL the majority of other tasks, e.g., the communication interface, resource management, and monitoring. In the following subsections, we detail how XVL addresses the limitations and missing features that we identified in Section II. Namely, how XVL: provides a communication interface for resource negotiation and allocation, supports the creation and destruction of virtual radios on demand, ensures their computational isolation, and allows the monitoring of their performance.

A. Resource Negotiation and Dynamic Allocation

The XVL operates based on a client-server paradigm, with a single server, i.e., one instance of XVL on top of a radio hypervisor, serving multiple clients, i.e., verticals and MVNOs. The clients can send messages to the server for querying information about the use of resources and requesting resources at any time. Figure 1 depicts the process of negotiating and using resources through XVL. Upon successful negotiation and reservation of resources, XVL interfaces with the radio hypervisor for creating/destroying the RAN slice. XVL has callbacks for informing the radio hypervisor about changes in the resource mapping and/or allocation of virtual radios, so the radio hypervisor can instantiate a new virtual RF front-end. Once the radio hypervisor completes the creation of the virtual RF front-end, XVL can perform additional configurations in the virtual radio (to be further discussed in Section III-B).

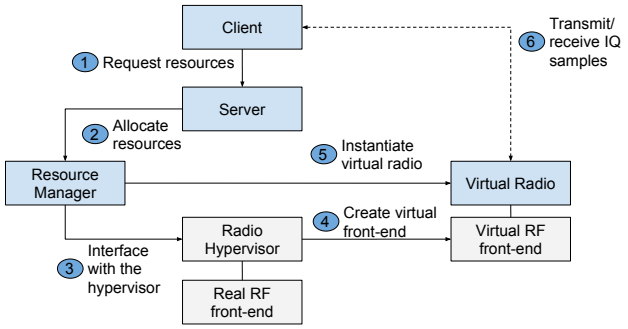


Fig. 1: An example of the steps involved for negotiating the use of resources and allocating a new virtual radio. XVL (blue) stands as a wrapper around the radio hypervisor (grey).

Finally, XVL informs the client of the proper ways to reach its virtual radio, e.g., through CPRI, OBSAI, or a UDP socket, from which the clients can start transmitting or receiving IQ samples.

At present, XVL supports virtualisation in the frequency domain. However, XVL separates the resource management from the underlying technicalities of the radio hypervisor, which allows the resource manager to be extended and modified at ease. Thus, XVL can easily be extended for supporting time or space instead of frequency resources, or even incorporate time and space as new degrees of freedom in the resource definition. It only depends on the types of radio virtualisation that the radio hypervisor supports. Currently, the radio resources are defined in terms of a centre frequency, a bandwidth, a client ID, and a UDP port.

B. Independence Between Virtual Radios

Every virtual radio has a virtual RF front-end, which the virtual radio employs for transmission and/or reception. Each virtual RF front-end uses a fraction of the total radio resources of the real RF front-end. The radio resources can be defined in terms of bandwidth, timeslot, or antennas, depending on the radio hypervisor's approach for virtualising the real RF front-end. Regardless of the type of radio resource, each virtual radio must send an appropriate number of IQ samples to the radio hypervisor at precise timing. The radio hypervisor consumes the IQ samples of all virtual radios at once and multiplexes the virtual RF front-ends into the real RF front-end. The number of necessary IQ samples per virtual radio, however, varies depending on the amount of radio resources per virtual RF front-end. In the case of an

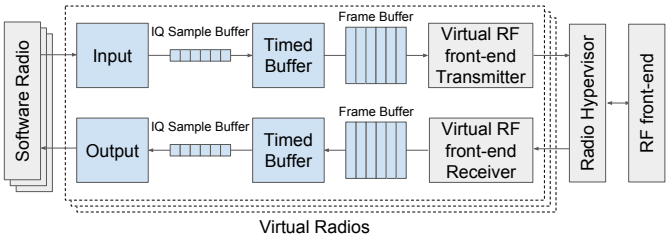


Fig. 2: The timed buffer consumes the received IQ samples and generates the windows that the radio hypervisor requires. The same process happens in the opposite direction.

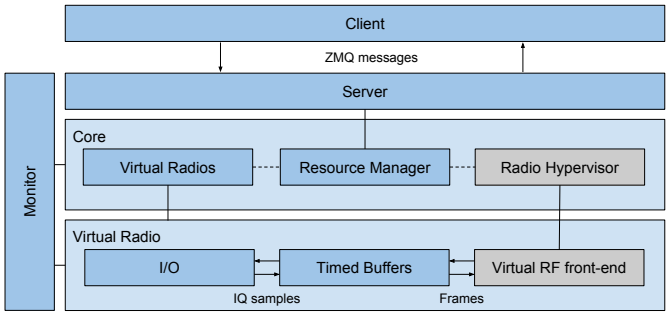


Fig. 3: Block diagram of XVL's architecture, the elements that compose it (blue), and the elements it must interface with (grey).

FFT-based hypervisor, e.g., HyDRA, the bandwidth of each virtual RF front-end is mapped onto a number of FFT bins. The virtual radios must generate a number of samples equal to the number of FFT bins to create an FFT window. Then, the radio hypervisor multiplexes the windows of the virtual RF front-ends together using an IFFT [17].

XVL employs timed buffers to ensure that the radio hypervisor receives the appropriate number of IQ samples at the right moment. Based on the overall sampling rate of the virtual radio and the hypervisor, the timed buffers output IQ samples at the precise moment for the radio hypervisor's consumption. Figure 2 illustrates the operation of timed buffers inside a virtual radio, where UDP sockets are used for receiving/sending IQ samples from/to remote software radios. In the case of overflows, XVL's internal timed buffer will start filling, until reaching a cap. After that, incoming samples are dropped to prevent the XVL process from overflowing memory. In the case of underflows, XVL can either pad the frames with missing IQ samples with zeroes or transmit empty frames while waiting for the missing samples. This way, the radio hypervisor and the rest of the system will not halt waiting for the delayed or missing IQ samples of a given virtual radio.

The virtual radios can either operate as transmitters, receivers, or transceivers. However, the virtual radios may not be symmetrical in both directions, i.e., they may employ different RATs and require different amounts of radio resources. For that reason, every virtual radio in XVL possesses completely independent transmitter and receiver chains. Figure 2 shows an example of a virtual radio with both types of chains: the transmit chain (top) receiving samples from a given software radio, and the receive chain (bottom), sending samples towards a given software radio. The radio hypervisor may realise each type of chain in different manners, i.e., different virtualisation approaches, or may use different radio hardware, i.e., different physical devices for transmission and reception.

C. Architecture and Monitoring Capabilities

Figure 3 illustrates the overall architecture of the XVL implementation. The clients interface with the server for resource discovery and negotiation through ZMQ messages. The server queries the resource manager and tries to fulfil requests. The resource manager has a list of virtual radios and can interact with the radio hypervisor for creating or destroying virtual RF front-ends. Each virtual radio has an Input/Output (I/O), a timed buffer and a virtual RF front-end, which is tied to the radio hypervisor. Aside from the functionality for supporting services, XVL also has a monitor utility for inspecting the KPIs of the virtual radios.

Currently, the monitoring utility is used for assessing the use of computational resources per virtual radio. It measures the buffer sizes, the sampling rates, and the time each virtual radio has to fill a frame. Based on the buffer sizes, it is possible to determine whether the software radio is presenting overflows, i.e., buffer sizes increasing steadily, or underflows, i.e., buffer sizes frozen at zero. We plan to extend this mechanism to evaluate the introduced SINR degradation and delay.

IV. Experimental Evaluation

In this section, we assess the performance of XVL in different scenarios. We evaluate XVL regarding the delay for provisioning virtual radios, the delay introduced in the communication pipeline, and the computational overhead. It is worth mentioning that we perform all of the aforementioned experimental analysis on a server equipped with an Intel Xeon E5-2620 v2 and 4GB of RAM.

A. Integration with a Radio Hypervisor and GNU Radio

As discussed in Section III, XVL is a software layer that sits on top of radio hypervisors and provides them with the missing functionality for supporting RANaaS. Hence, XVL requires the integration with an underlying radio hypervisor to operate. Based on the literature review and analysis in Section II-B, we opted for using HyDRA [17] as the underlying radio hypervisor, due to HyDRA being platform agnostic and providing an open-source prototype. We integrated XVL with HyDRA for validating XVL's operation and showcasing its features. As a result of the integration with XVL, we enabled HyDRA to support RANaaS. Furthermore, we developed a client-side library for communicating with XVL. This library allows any C++-based client to interface with XVL for querying, requesting, and using XVL's resources. We employ ZMQ, a cross-platform networking library, which facilitates porting our communication library to serve clients implemented in other languages, e.g., Python, Java, or Ruby. We used this library for developing GNU Radio blocks that can seamlessly replace a USRP source/sink block in existing GNU Radio

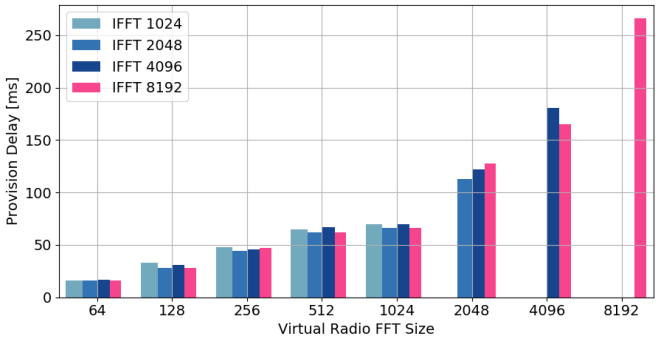
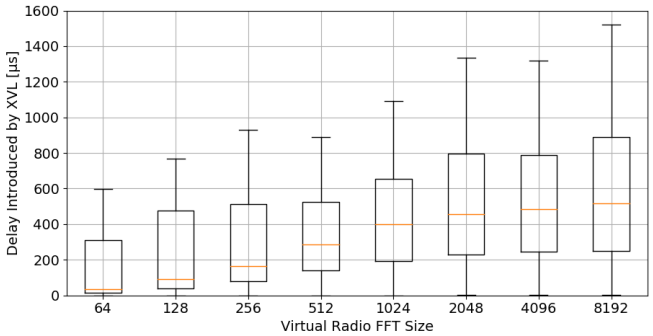
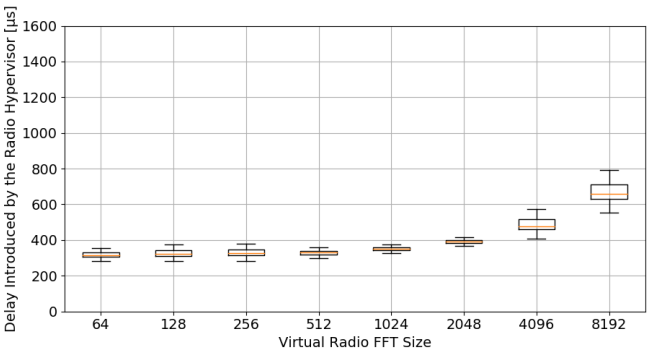


Fig. 4: The provisioning delay to fulfil a virtual radio request. The time taken for creating virtual radios is proportional to the number of allocated FFT bins, but not to the overall IFFT size of the radio hypervisor.



(a) The service delay introduced by XVL. The service delay increases with the number of resources, due to the longer time required to allocate larger frames in memory.



(b) The service delay introduced by HyDRA. The service delay increases with the number of resources, due to the increasing complexity of the FFT/IFFT operations.

Fig. 5: Delay measurements for different virtual radio configurations. All the measurements were made using a real bandwidth of 2 MHz and an overall FFT size of 8192 bins, which corresponds to the worst case scenario in Section IV-B.

flowgraphs. These blocks only require the extra information of the IP address and port number of where an XVL instance is running, and the client ID. The benefits of this setup are twofold: it allows us to use an existing radio hypervisor and GNU Radio flowgraphs for evaluating XVL, and it enables any GNU Radio flowgraph to leverage XVL for instantiating its virtual RF front-end.

B. Provisioning Delay

In this analysis, we are interested in the delay for provisioning virtual radios with XVL. The provisioning delay is crucial for planning and evaluating a RAN deployment using the RANaaS paradigm, as it dictates the time interval required to provision the virtual radios that realise the RAN. This interval comprehends the time between the client sending a resource request, and XVL returning the resource allocation confirmation (there are several procedures that occur between these two events, as we have seen in Section III). Figure 4 shows the results of our measurements for different configurations of the virtual radio and radio hypervisor. The exponential growth in the semi-log scale denotes a linear behaviour in relation to the number of resources allocated per virtual radio, regardless of the FFT size of the radio hypervisor. We attribute the linear behaviour to the time for allocating memory for XVL's buffers. The worst case scenario where a virtual radio required 8192 FFT bins took less than 300ms to be fulfilled. Hence, this result indicates that XVL is fast enough to provision virtual radios on-the-fly.

C. Service Delay

In this analysis, we are interested in the delay that XVL and HyDRA introduce to serve virtual radios and forward their IQ samples. The service delay is crucial to analyse the impact caused by using virtualised infrastructure instead of a real one, and to evaluate whether it can impair the operation of the virtual radios. First, we measured the time spent by XVL for delivering the IQ samples to the radio hypervisor, i.e., from successful UDP receptions, through the timed buffers, and generation of a frame. Figure 5a shows the results of our measurements, with a median of 343μs. Then, we measured the time spent by HyDRA to virtualise the RF front-end, i.e., consume the frames, multiplex the virtual RF front-ends using an IFFT, and forward the IQ samples to the real RF front-end. Figure 5b shows the results of our measurements, with a median of 334μs. These results show that XVL introduces a service delay within the same order of magnitude of the radio hypervisor. However, the overall median introduced delay by using XVL and HyDRA for supporting RANaaS falls under 1ms. A 1ms delay can be impactful in the delay budget for Centralised-RAN (C-RAN) scenarios, and must be taken into account in the network planning.

D. Computational Overhead

In this analysis, we consider the computational overhead introduced by running XVL alongside a radio hypervisor to serve LTE and NB-IoT virtual radios. XVL and the radio hypervisor introduce a new layer of complexity between the software radios and the RF front-end, with an overhead that must be quantified for assessing whether it is feasible to use XVL. We measured the CPU utilisation for an XVL and HyDRA instance for different IFFT sizes and normalised it as a percentage of one CPU core. Figure 6 shows the results of our measurements. Independently of the configuration used in the radio hypervisor, the combination of XVL and HyDRA,

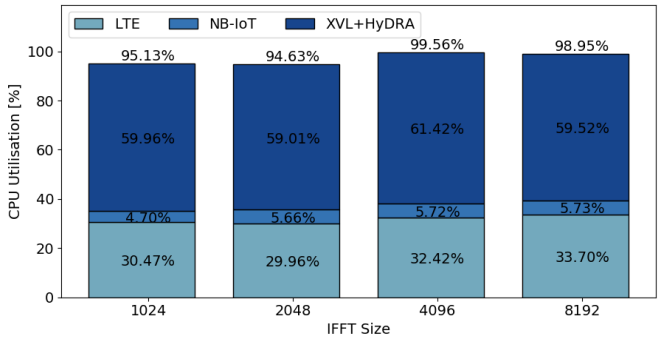


Fig. 6: CPU utilisation for running XVL alongside HyDRA, in addition to an LTE and an NB-IoT virtual radios, under different radio hypervisor configurations.

the NB-IoT, and the LTE, require roughly 60%, 5% and 30% of the CPU processing power, respectively. The constant CPU utilisation by the virtual radios is expected as their processing is independent of the radio hypervisor. However, we did not capture any increase in the CPU overhead for the radio hypervisor as shown in [17]. We attribute this to the multithreading that XVL employs for managing the timed buffers and sockets, decreasing the significance of the CPU utilisation of the radio hypervisor. These results indicate that XVL can run alongside a radio hypervisor on commodity hardware for deploying multiple virtual radios.

V. Conclusion

In this paper, we addressed the missing functionality for radio hypervisors to support RANaaS. We compiled a comprehensive list of features necessary to enable a radio hypervisor to provide RANaaS, and we evaluated the current state-of-the-art with respect to these requirements. Then, we presented XVL, a software layer that can be added on top of existing hypervisors and provides them with the missing capabilities for using the hypervisors to support RANaaS. We evaluated XVL regarding its provisioning delay, service delay, and computational overhead. Our results show that XVL provisions RAN slices in real-time, introduces a delay comparable to a radio hypervisor, and can run on commodity hardware. We showed that the combination of XVL with a radio hypervisor enables the use of SDRs for RANaaS. Moreover, we made available to the community the source code of the implementation of XVL in the public repository: https://bitbucket.org/joaofelipesantos/xvl/. We are continuing to extend XVL and introduce new features, e.g., the visualisation of waveforms on the virtual and real RF front-ends, and the provisioning of radio functionality, e.g., modulation and coding, on top of the virtual radios.

Acknowledgements

The research leading to this letter received funding from the European Horizon 2020 Program under the grant agreements No. 732174 (ORCA project) and No. 732497 (5GINFIRE).

References

[1] 3rd Generation Partnership Project, "3GPP TR 22.891: Feasibility Study on New Services and Markets Technology Enablers," 3rd Generation Partnership Project, Tech. Rep., Sep. 2016.

[2] ——, "3GPP TR 28.801: Study on Management and Orchestration of Network Slicing for Next Generation Network," 3rd Generation Partnership Project, Tech. Rep., May 2017.

[3] 5G America, "5G Americas White Paper – Network Slicing for 5G and Beyond," 5G America, Tech. Rep., 2016.

[4] X. Zhou *et al.*, "Network Slicing as a Service: Enabling Enterprises' Own Software-Defined Cellular Networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.

[5] 3rd Generation Partnership Project, "3GPP TR 38.801: Study on New Radio Access Technology: Radio Access Architecture and Interfaces," 3rd Generation Partnership Project, Tech. Rep., Mar. 2017.

[6] J. F. Santos *et al.*, "Orchestrating Next-Generation Services Through End-to-End Network Slicing," White Paper, The ORCA Consortium, Oct. 2018. [Online]. Available: https://orca-project.eu/wp-content/uploads/sites/4/2018/10/orchestrating_e2e_network_slices_Final.pdf

[7] T. Taleb *et al.*, "EASE: EPC as a Service to Ease Mobile Core Network Deployment Over Cloud," *IEEE Network*, vol. 29, no. 2, pp. 78–88, 2015.

[8] D. Sabella *et al.*, "RAN as a Service: Challenges of Designing a Flexible RAN Architecture in a Cloud-Based Heterogeneous Mobile Network," *IEEE Future Network & Mobile Summit*, 2013.

[9] R. Sherwood *et al.*, "FlowVisor: a Network Virtualization Layer," *OpenFlow Switch Consortium, Tech. Rep*, vol. 1, p. 132, 2009.

[10] A. Al-Shabibi *et al.*, "Openvirtex: A Network Hypervisor," in *Open Networking Summit (ONS)*, 2014.

[11] J. van de Belt *et al.*, "Defining and Surveying Wireless Link and Network Virtualization," *IEEE Communications Surveys & Tutorials*, 2017.

[12] J. Sachs and S. Baucke, "Virtual Radio: A Framework for Configurable Radio Networks," in *ACM Conference on Wireless Internet (WICON)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, p. 61.

[13] C. Liang and F. R. Yu, "Wireless Network Virtualization: A Survey, Some Research Issues And Challenges," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 358–380, 2015.

[14] M. Richart *et al.*, "Resource Slicing in Virtual Wireless Networks: A Survey," *IEEE Transactions on Network and Service Management*, vol. 13, no. 3, pp. 462–476, 2016.

[15] H. Wen *et al.*, "Current Trends and Perspectives in Wireless Virtualization," in *IEEE International Conference on Selected Topics in Mobile and Wireless Networking (MoWNeT)*, 2013, pp. 62–67.

[16] F. Fu and U. C. Kozat, "Wireless Network Virtualization as a Sequential Auction Game," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2010, pp. 1–9.

[17] M. Kist *et al.*, "SDR Virtualization in Future Mobile Networks : Enabling Multi-Programmable Air-Interfaces," in *IEEE Internationl Conference on Communications (ICC)*, May 2018.

[18] R. Kokku *et al.*, "NVS: A Substrate for Virtualizing Wireless Resources in Cellular Networks," *IEEE/ACM Transactions on Networking*, vol. 20, no. 5, pp. 1333–1346, 2012.

[19] L. Xia *et al.*, "Virtual WiFi: Bring Virtualization from Wired to Wireless," in *ACM SIGPLAN Notices*, vol. 46, no. 7, 2011, pp. 181–192.

[20] X. Foukas *et al.*, "Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture," in *ACM International Conference on Mobile Computing and Networking (MobiCom)*, 2017, pp. 127–140.

[21] K. Tan *et al.*, "Enable Flexible Spectrum Access with Spectrum Virtualization," in *IEEE Dynamic Spectrum Access Networks (DYSPAN)*, 2012, pp. 47–58.

[22] E. Blossom, "GNU Radio: Tools for Exploring the Radio Frequency Spectrum," *Linux journal*, vol. 2004, no. 122, p. 4, 2004.